ACEEE Int. J. on Recent Trends in Engineering and Technology, Vol. 10, No. 1, Jan 2014

Modeling SYN Flooding DoS Attacks using Attack Countermeasure Trees and Finding Optimal Set of Countermeasure using a Greedy Algorithm

Dr. Kalpana Yadav[1], Rupam Mittal[1], Priya Goel[1] and Sahaj Biyani[2]
[1] Indira Gandhi Institute of Technology, Delhi, India
Email: {kyadav11, rupam.mittal, priyagoel.1254}@gmail.com
[2] Delhi Technological University, Delhi, India
Email: sahajbiyani@gmail.com

***Abstract*— In this paper, a greedy algorithm is proposed, to find optimal set of countermeasures that minimizes total cost of Security Investment subject to constraint that it covers entire set of attack events. This algorithm makes use of Birnbaum's Structural Importance Measure to find compute criticality of basic attack events in achieving the goal. It helps in prioritizing the countermeasures covering the attack events.**

***Index Terms*— Birnbaum's Structural Importance Measure, ROI, ROA, Attack Countermeasure Tree**

I. INTRODUCTION

Our society depends on a wide variety of telecommunication services to support our demands for everything from pure entertainment to commerce, banking and life critical services. Critical services such as transport traffic control systems, and emergency and financial services, have stringent QoS requirements for both performance tolerance and service dependability. Non-critical services like entertainment will typically have less strict requirements.

Various different types of threats like attacks, accidents, failures, natural disasters will cause service degradation or at least interruption of critical services. With increasing number of attacks on an organisation, it has become extremely important to identify, assess and mitigate the probable risks. Thus, security investment is essential to any enterprise. Goal is to make security managers aware of all the possible risk on organisation's asset to aid them in selecting the countermeasures that can bring the risk under acceptable levels.

Often determination of acceptable risk levels corresponding to various attacks and selecting best set of countermeasures to counteract them is not a one day's task. Most complex algorithms and modelling techniques, all have been designed and developed just to assist security managers of enterprises, in selection of best techniques, to protect their information infrastructure from interception by foreign agents. First step in the process is security modelling, which aims at designing a scalable model that helps in visualizing the attack scenario as well as in finding the security solution.

Attack trees help in figuratively understanding the various ways to attack a system. Disadvantages of it are that it represents only the attacker's perspective and has no mechanism for inclusion and analysis of defence strategies along with various attacks. Thus the standard attack trees are augmented with set of

DOI: 01.IJRTET.10.1.532
© Association of Computer Electronics and Electrical Engineers, 2014

countermeasures. Countermeasures corresponding to a particular path in the tree, exploiting a specific vulnerability, are added at the leaf node. This modified version of attack trees are called as Defence Trees. Disadvantages of defence trees are that it places defence mechanisms only at the leaf nodes. Although attacks and defences are both accounted for in Attack response trees but it suffers from state space explosion problem.

Thus a non-state space model for modelling attacks as well as countermeasures (in form of detection and mitigation events) together was formulated and called as Attack Countermeasure Trees or ACT. In ACT, defence mechanisms can be placed at any node, attack scenarios and attack-countermeasure scenarios. Both probabilistic and qualitative analysis (not involving probabilities) can be performed.

The algorithm presented in this research paper performs a qualitative analysis of ACT using Birnbaum's Structural importance measure and finds optimal set of countermeasure using it.

An attempt by a malicious (or unwitting) user, process, or system to prevent legitimate users from accessing a resource (usually a network service) by exploiting a weakness or design limitation in an information system is called Denial of Service(DoS) attacks. A SYN Flooding attack is a form of denial-of-service attack in which an attacker sends a succession of TCP connection requests to a target's system in an attempt to consume enough server resources to make the system unresponsive to legitimate traffic.

A case study of SYN Flooding attacks modelled using ACT has been taken in this paper to determine the optimal set of countermeasure for the case by using the new algorithm proposed.

II. Related Work

Schneier developed the basic attack tree (AT) formalism [3]. Dewri *et.al* [4] used genetic algorithms to find optimal security action using single and multi-objective optimization on ATs. However these methods suffered because AT structure did not take countermeasures into account. Bistarelli *et.al* [6] proposed defence trees (DTs) to incorporate defence mechanisms in AT but only at leaf nodes and used ROI and ROA measures to find best countermeasure. Bistarelli *et.al* [5] then applied game theory to find the most cost effective set of countermeasures. However in DTs, countermeasures could only be placed at the leaf nodes. Zonouz *et.al* [9] proposed attack-response trees (ARTs) that incorporate both attacks and responses but they used a state-space model (partially observable stochastic game model) to find an optimal set of countermeasures thereby leading to state-space explosion problem. Attack Countermeasure Tree (ACT) [2] provides a simple yet compact approach for determination of security policy, harnessing the benefits of the aforementioned models and also allowing to perform optimal countermeasure selection for different attack scenarios under given constraints with a non-state-space approach. Arpan roy *et.al* [1] proposed algorithms for optimal countermeasure selection with as well as without probability assignments. But algorithms without probability assignments were based on determination of mincuts that represented various attack countermeasure scenario. On contrary, the algorithm presented in this paper makes use of Birnbaum's Structural Importance Measure [1] to design a greedy algorithm that also takes into account relative critical importance of attack events and then determines optimal set of countermeasures with minimum security investment cost.

III. Background

A. Birnbaum's Structural Importance Measure

In general, importance ranks the components in a system using a numerical rank (relative importance), based on certain system characteristic of interest, such as the component's contribution to a system (failure) event occurrence. Birnbaum first introduced the concept of importance in 1969 [8], and one of the most widely used reliability importance indices is Birnbaum's component importance [8]. Consider a multi-component system with k independent equally probable components. Analytically [7] this is defined by

$$I_k^B(t) = \frac{\partial S(t)}{\partial R_k(t)} \qquad (1)$$

Where,

$I_k^B(t)$ = Structural Importance Measure of k^{th} component.

$S(t)$ = Reliability of entire system at time t (Probability that system will be functional in a given state)

$R_k(t)$ = Reliability of k^{th} component at time t (Probability that the component will be functioning at time t)

Relationship between reliability and failure probability, P_F of a component is explained now.
A component can even be the entire system.

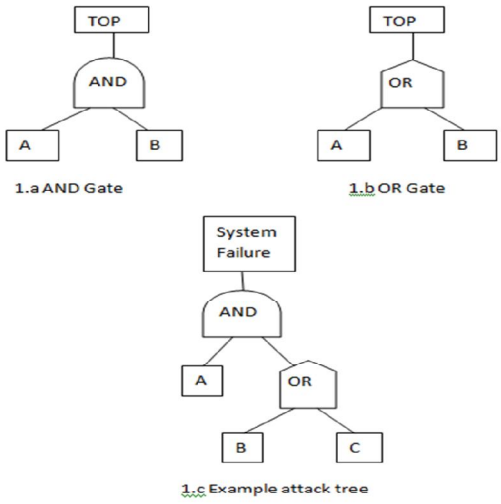

Figure 1. Example attack tree

Reliability is probability that component will function as expected, i.e. successfully, so

$$R = \frac{Success}{Success+Failure} \quad (2)$$

P_F is the probability that given component will fail. Hence,

$$P_F = \frac{Failure}{Success+Failure} \quad (3)$$

Therefore,

$$R+P_F = 1 \text{ , i.e. } R = 1-P_F \quad (4)$$

Hence, equation (1), can also be expressed as below, giving I_k^B in terms of failure probability,

$$I_k^B(t) = \frac{\partial P_F(SYS)}{\partial P_F(k)} \quad (5)$$

In our case of ACT, where we want to use this concept to evaluate critical importance of attack events, the components considered are various attack events in the corresponding Attack tree. If we know the failure probabilities of basic independent events that are fed into a particular gate, we can calculate the P_F of top event. The event at output of that gate as follows,

OR GATE
From fig (1.a), failure of either of two independent events A and B will cause System Failure,

$$P_F(SYS) = P_F(AUB)$$
$$P_F(SYS) = P_F(A) + P_F(B) - P_F(A)P_F(B) \quad (6)$$

AND GATE
From fig (1.b), both of two independent events A and B should fail to cause System Failure,

$$P_F(SYS) = P_F(A \cap B)$$
$$P_F(SYS) = P_F(A)P_F(B) \quad (7)$$

Let us consider an attack tree given in fig (1.c), all basic attack events [A, B and C] are independent and hence failure probability of each can be considered equal to 0.1.

Failure Probability for entire system,

$$P_F(SYS) = P_F(TOP)$$
$$= P_F(A)P_F(G)$$
$$= P_F(A)[P_F(B) + P_F(C) - P_F(B)P_F(C)]$$
$$= P_F(A)P_F(B) + PF(A)P_F(C)\ P_F(A)P_F(B)P_F(C)$$

$$I_A^B(t) = \frac{\partial P_F(SYS)}{\partial P_F(A)}$$
$$= P_F(B)+P_F(C) - P_F(B)P_F(C)$$
$$= 0.19$$

$$I_B^B(t) = \frac{\partial P_F(SYS)}{\partial P_F(B)}$$
$$= P_F(A)[1-P_F(C)]$$
$$= 0.09$$

$$I_c^B(t) = \frac{\partial P_F(SYS)}{\partial P_F(A)}$$
$$= P_F(A)[1 -P_F(C)]$$
$$= 0.09$$

B. Attack Countermeasure Trees

In ACT[2], there are three distinct classes of events: attack events (e.g., install keystroke logger), detection events (e.g., detect keystroke logger) and mitigation events (e.g., remove keystroke logger). Figure 2 shows most common case of an ACT with a single attack event A and multiple pairs of Detection [D_i] and mitigation events [M_i]. Probability of attack success at top node, $P_{GOAL}$is given below,

$$P_{GOAL} = \prod_{i=1}^{n}(1 - p_{Di} + p_{Di}\ (1 - p_{Mi}))$$
$$= p_A(1 - p_{Di}X\ p_{Mi})) \qquad (8)$$

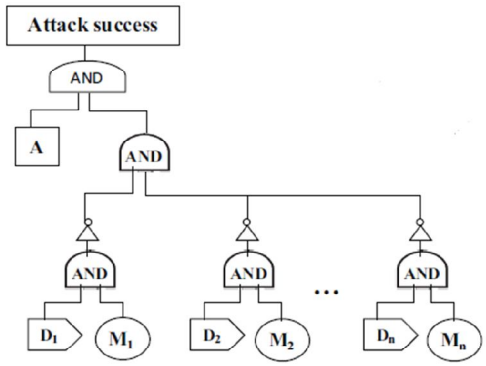

Figure 2. ACT with one attack and multiple pairs of detection and mitigation events

Quantitative as well as qualitative analysis can be performed using ACT. Quantitative analysis can be performed using mincuts[8] and structural importance measure[6]. In ACT, the top event is associated with the set of all mincuts. Mincuts of an ACT represent attack-countermeasure scenarios. Structural Importance Measure Analysis which is important to determine the most critical event in ACT is used when ACT has equiprobable events, i.e., we are provided with only the ACT but probability of attack (for attack events) and detection/mitigation (for detection/mitigation events) are unknown.

For ACT, the probability of a successful attack can be computed which can be further split into the probability that the attack is undetected and the probability that the attack is detected but unmitigated. When provided with values for parameters such as probabilities of attacks, cost *etc.*, probabilistic (or quantitative) analysis can be performed using ACTs. Quantitative analysis using ACT can be viewed from two distinct viewpoints: attackers' viewpoint and defender's (or security analyst's) viewpoint The measures such as

attack cost and ROA reflect the attacker's perspective whereas the metrics such as security investment cost, risk, impact and ROI represent the defender's perspective.

IV. OPTIMAL COUNTERMEASURE SELECTION WITHOUT PROBABILITY ALGORITHM

It has been found that it is very difficult or in certain cases impossible to find out the probability of attack, detection and mitigation events, especially for research purposes. This is because these events correspond to potentially dangerous as well as rarely occurring events in real world scenario. Hence, finding probability values for them is a tough task.

Therefore, the algorithm proposed here, to find out optimal set of countermeasures does not require probability assignments to be made to event in an ACT. Initially, the Structural Importance Measure, of all the basic Attack events, attack events at the leaf level of ACT are determined. For this, equal values of Failure Probability to each such attack event are assigned and then its Importance Measure is computed using equation (5). Based on the values of Importance Measure, the attack events are ordered in decreasing order of their criticality value. Highly dangerous and probable event will have high value of Importance Measure and vice versa. Now the attack event with highest value of I_κ^B is picked up and the set countermeasure covering that attack event is found. If the set consists of only one element, that CM (countermeasure) is added to OPT (optimal set of countermeasure). If, however, the set has more than one element, the countermeasure with minimum security investment cost is found and added to OPT. It is based on a rationale that it is most important to include countermeasure, CM for highly critical attack event. All the attacks covered by CM are then found and removed from total set of attacks and their Importance Measure from I, as they have already been covered. This helps to minimize size of OPT. The process is continued in this manner, taking attack events with decreasing values of Importance measure successively and adding corresponding countermeasure to OPT until all attacks have been covered.

Formal Algorithm to find optimal set of countermeasures based on relative importance of attack events being covered by them and at the same time minimising Security Investment cost is as follows:

Some common notations are:

$S(A) = \{ A1, A2,\ldots,A_m\}$ set of m attack events
$S(C) = \{C1,C2,\ldots C_n\}$ set of n countermeasures,
where each countermeasure represents a pair of detection and mitigation events in ACT, i.e. Cj = (Dj, Mj).
$P_F(A)$ = Failure Probability of attack event A
SIC [1: n] = Array containing Security Investment Cost of countermeasures
M_{ij} = Attack event A_i is covered by countermeasure C_j
I [1:m] = Array containing Importance measures of attack events
COUNTER = Set of countermeasures that cover a given attack event
INPUT: MINC = Set of mincuts obtained from SHARPE.
OUTPUT: C_{OPT} = Set of optimal countermeasures
FIND_ATTACKS_COVERED(C, MINC) is a function that takes a countermeasure C and finds corresponding set of attacks covered by it.

Following is the pseudo code of the proposed algorithm:

```
OPT_C (MINC)
{
    1. Determine S(A) and S(C) from MINC
       Set C_OPT ←{φ}
       Set m ←|S(A)|, n ←|S(C)|
       for i = 1 to m
         Set P_F(A_i) ←0.1
       end for
       for j = 1 to n
         SIC[j] ←Security Investment Cost of countermeasure C_j
       end for
```

2. Create a m by n matrix
   for I = 1 to m
      for j = 1 to n
        Set M_{ij} ← 0
      end for
   end for
   for each C_j ∈ S(C)
      COV_ATT ← FIND_ATTACKS_COVERED(C, MINC)
      for each Ai ∈ COV_ATT
         Set M_{ij} ← 1
      end for
   end for

3. Convert ACT to Attack Tree by deleting detection and mitigation events so that only attack events remain.

4. Determine expression for P_F (SYS) by starting from leaf nodes and moving upwards towards goal nodes, calculating value of failure probability at output of gates at each step.

5. For k = 1 to m
   Determine structural importance measure for A_K using
   $I_A^B(t) = \frac{\partial P_F(SYS)}{\partial P_F(K)}$
   Add I_k^B to k^{th} location of array I [1: m]
   End for

6. Sort array I in decreasing order in such a way that Importance measure of an attack in I and corresponding attack event in S(A) are at same position in respective arrays.

7. i ← 1
   While S(A) != φ
   {
    A ← Attack with I[i]
    For A_k = A
      For j = 1 to n
        If M_{ij} = = 1
        COUNTER ← C_j
      End for
    End for
    If |COUNTER| = = 1
      OPT ← CM
    Else
       Find countermeasure, CM in COUNTER with minimum Security Investment cost stored in array SC and add it to OPT.
    COV_ATT ← FIND_ATTACKS_COVERED(CM, MINC)
    S(A) ← S(A) – COV_ATT

    For each A ∈ COV_ATT
       Remove Importance measure of A from array I
    End for
   }

   FIND_ATTCKS_COVERED(C, MINC)
   {

```
A (Set of attack events covered by Cj)  ←{φ}
Do
{
   Traverse a mincuts I containing C
   For each attack Ac ε I
       A ←A U Ac
  End for
}
while (all mincuts have been traversed)
return A
                                        }
```

V. Modeling Syn Flooding Attack

A SYN Flooding attack is a form of denial-of-service attack in which an attacker sends a succession of TCP connection requests to a target's system in an attempt to consume enough server resources to make the system unresponsive to legitimate traffic.

Normally, as shown Figure 3.a, when a client attempts to start a TCP connection to a server, the client and server exchange a series of messages. Normally, the client requests a connection by sending a SYN (*synchronize*) message to the server and the server acknowledges this request by sending SYN-ACK back to the client. The client responds with an ACK, and the connection is established. This is called the TCP three-way handshake, and is the foundation for every connection established using the TCP protocol.

The backlog queue is a large memory structure that allocates part of the system memory called as Transmission Control Block (TCB) to store the state of the connection by copying some header field values of SYN packet to TCB. The backlog queue controls how many half-open connections can be handled by the operating system at the same time. When this limit is reached, subsequent requests are silently dropped by the operating system.

In case of SYN Flooding attack (Fig 3.b), where attacker will bombard the victim machine with a large number of TCP connection requests trying to create a large number of half open connections. The victim sends SYN+ACK for every SYN packet received. The attacker can simply not reply with an ACK message or it might spoof source IP addresses of the SYN' senders. Either ways, number of half open connections keep building up and using the victim machine's resources such as backlog and system memory. Due to this, legitimate connection requests are denied. This leads to Denial of Service Attack. SYN Flooding attack can either be performed from a single machine as was done when it originally started in 1994 or from a group of compromised machines, called together as a botnet.

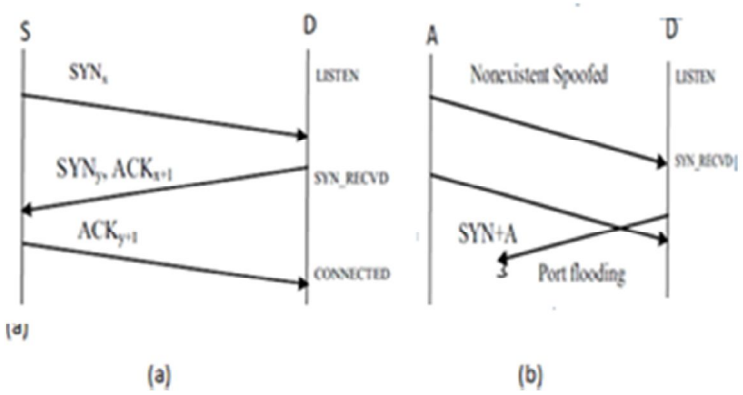

Figure 3 (a) Three-Way Handshake (b) System under attack

A. Syn Flooding Attack From A Single Attacker

SYN flooding attack when launched from a single machine is usually done with the help of scripts like Juno.c, Juno_z101f.c, Dos.pl, SYNflood or by using tools called as attack generators that are made to launch

multiple TCP connection establishment requests to the attacker in short amount of time. The source IP address in these connection requests (SYN packets) can either be of the attacker or can be spoofed. For both cases, we can just use Traceroute to track the source, i.e. the attacker. Traceroute is a computer network diagnostic tool for displaying the route (path) and measuring transit delays of packets across an Internet Protocol (IP) network. The traceroute command is available on a number of modern operating systems. We can then deploy a firewall at the last mile router to block the requests from that specific host to mitigate the attack. Two cases and their corresponding countermeasures are discussed below:

(a) Non-spoofed source IP Address: when the source IP address in the SYN packets sent for the attack is of the attacker itself as was done in the initial SYN Flooding attacks, it is easy to detect the attack when too many requests in half-open state are coming from the same source. So Traceroute can be used to track the source and then deploy a firewall.

(b) Spoofed IP Address: When source IP Address of various connection requests is set to different unrelated values to given an effect that all those requests are coming from different hosts and thus are legitimate. This is what makes SYN flooding attacks even difficult to detect.

Spoofing techniques can be categorized into different types according to what spoofed source addresses are used in the attacking packets. The three common IP spoofing types are random spoofing, subnet spoofing, and fixed spoofing **.** In random spoofing, the attacker randomly generates 32-bit numbers for use as source addresses of the attacking packets. In subnet spoofing, the addresses are generated from the address space corresponding to the subnet in which the agent machine resides. For example, a machine which is part of the 143.89.124.0/24 network may spoof any address in the range from 143.89.124.0 to 143.89.124.255. Another type of IP spoofing, called fixed spoofing, chooses source addresses from a given list. In this case, the attacker typically wants to perform a reflector attack or impose a blame for attack on several specific machines.

The technique that is used to detect SYN flood attack in case spoofing is used is based on the storage-efficient data structure, which is a variant of Bloom filter, is used to generate a hash digest of the traffic. The change-point detection method is based on the CUSUM algorithm**,** which is a nonparametric change-point detection method. After some information about the traffic is extracted and stored in the Bloom filter, CUSUM is then applied to detect abnormal changes in the digested traffic. When traffic is detected as an attack we classify it into three type of spoofing. To protect our host against the attack once it has been detected, we use soft rate limiting scheme where we allow only a fraction of traffic to pass through. We can also use Traceroute to track the source and then deploy a firewall.

B. Attack From Botnet

A set of compromised hosts are used to launch the attack together, form the botnet. To detect such an attack , we can use netstat command. The netstat command shows us how many connections are currently in the half-open state. For example in figure 4, the half-open state is described as SYN_RECEIVED in windows and as SYN_RECV in UNIX systems.

```
# netstat -n -p TCP
tcp 0  0 10.100.0.200:21 237.177.154.8:25882     SYN_RECV
tcp 0  0 10.100.0.200:21 236.15.133.204:2577     SYN_RECV
tcp 0  0 10.100.0.200:21 127.160.6.129:51748     SYN_RECV
tcp 0  0 10.100.0.200:21 230.220.13.25:47393     SYN_RECV
tcp 0  0 10.100.0.200:21 227.200.204.182:60427   SYN_RECV
tcp 0  0 10.100.0.200:21 232.115.18.38:278       SYN_RECV
tcp 0  0 10.100.0.200:21 229.116.95.96:5122      SYN_RECV
tcp 0  0 10.100.0.200:21 236.219.139.207:49162   SYN_RECV
tcp 0  0 10.100.0.200:21 238.100.72.228:37899    SYN_RECV
...
```

Fig 4 Example of netstat command

To mitigate this kind of attack that exploits various vulnerabilities in the Operating system like size of backlog, time half open connection are kept in backlog etc, we make use of \strategies related to OS itself. They are described below:

(a) Increasing size of the backlog: Under a SYN attack, we can modify the backlog queue to support more connections in the half-open state without denying access to legitimate clients. Increasing the backlog queue size requires that a system needs additional memory resources for incoming requests. If a system has not enough memory for this operation, it will have a bad impact on system performance

(b) Decreasing SYN_RECEIVED time: When a server receives a request, it immediately sends a response with the SYN and ACK flags set, puts this half-open connection into the backlog queue and then waits for a packet with the ACK flag set from the client. When no response is received from the client, the server retransmits a response packet (with the SYN and ACK flags set) several times (depending on default value in each operating system) by giving the client a chance to send the ACK packet again. It is clear that when the source IP address of a client was spoofed, the ACK packet will never reach. After a few minutes the server removes this half-open connection. We can speed up this time of removing connections in the SYN RECEIVED state from the backlog queue by changing time of first retransmission and by changing the total number of retransmissions.

(c) Enabling SYN Cookies: SYN cookies allocate no state at all for connections in SYN-RECEIVED. Instead, they encode most of the state (and all of the strictly required) state that they would normally keep into the sequence number transmitted on the SYN-ACK. If the SYN was not spoofed, then the acknowledgement number (along with several other fields) in the ACK that completes the handshake can be used to reconstruct the state to be put into the TCB. Else if it was spoofed coming from a malicious host, TCB would not be stored and would not consume victim's resources.

Hence, Attack countermeasure Tree for SYN Flooding attack has been given in figure 5.

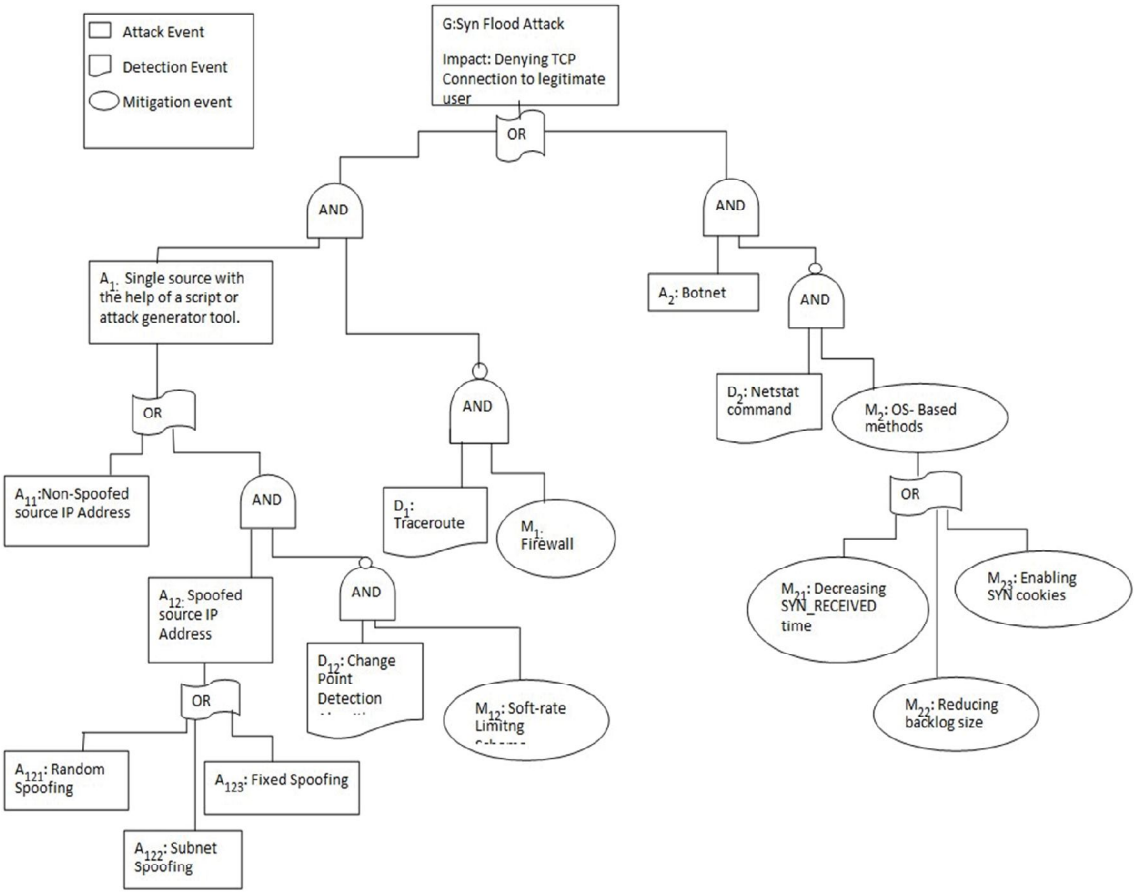

Figure 5. ACT for SYN Flooding DoS Attack

VI. Example Using The Proposed Algorithm

Let us apply the new proposed algorithm to ACT in order to find its optimal set of countermeasures.
Let us assign equal probability of failure to all the basic independent attack events i.e.
$P_F(A_{11}) = P_F(A_{121}) = P_F(A_{122}) = P_F(A_{123}) = P_F(A_2) = 0.1$
Mincuts for this ACT are:
(A_1,CM_1) , (A_{121},CM_1,CM_{12}) , (A_{122},CM_1,CM_{12}) , (A_{123},CM_1,CM_{12}) , (A_2, CM_2)
Where,
$CM_1 = (D_1, M_1)$
$CM_{12} = (D_{12}, M_{12})$
$CM_2 = (D_2, M_2)$
m (number of basic independent attack events) = 5
n (number of countermeasures) = 3.

1. S(A) = {a ,b, c, d, e,}
   S(C) = {1,2,3}
   C_{OPT} = {φ} ; m=5,n=3
   SIC[] = {15000, 20000,14000}
2. Now, we create matrix M of size 5X3, where $M_{ij} \leftarrow 1$ if A_i is covered by countermeasure C_j,

	CM_1	CM_{12}	CM_2
A_{11}	1	0	0
A_{121}	1	1	0
A_{122}	1	1	0
A_{123}	1	1	0
A_2	0	0	1

3. Now converting ACT in figure 6 to Attack tree, we have,

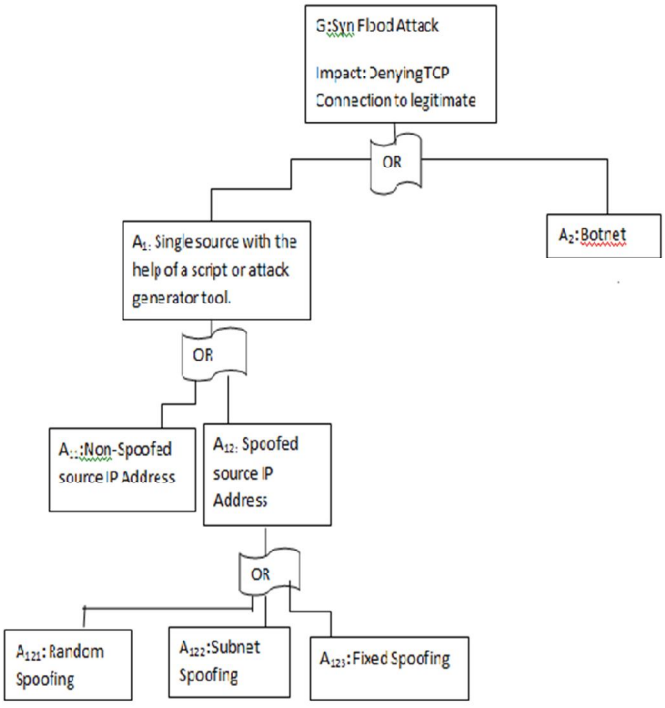

Figure 6. Attack Tree for SYN Flooding Attack

4. For simplicity let us denote $P_F(A_i)$ by A_i

$P_F(SYS) = [1-(1 - A_2)\{1 - (1 - A_{11})[1 - [1 - (1 - A_{121})(1- A_{122})(1 - A_{123})]]\}]$
$= [1 - (1 - A_2)\{1 - (1 - A_{11})[(1 - A_{121})(1 - A_{122})(1 - A_{123})]\}]$

Let $(1 - A_{121})(1 - A_{122})(1 - A_{123})$ be Z.
$P_F(SYS) = [1 - (1 - A_2) \{1 - [(1 - A_{11})(Z)]\}]$
$= [1 - (1 - A_2)(1 – Z + A_{11}Z)]$
$= Z - A_{11}Z + A_2 - A_2Z + A_2A_{11}Z$

Now,
$Z = - (1-A_{121}) (1-A_{122}) (1-A_{123})$
$= (1 - A_{121})(1 - A_{122} - A_{123} + A_{122}A_{123})$
$= 1 - A_{121} - A_{122} - A_{123} + A_{122}A_{123} + A_{121}A_{122} + A_{121}A_{123} - A_{121}A_{122}A_{123}$

So,
$P_F(SYS) = A2+Z(1+A2A11-A2-A11)$
$= A_2 + (1 - A_{121} - A_{122} - A123 + A_{122}A_{123} + A_{121}A_{122} + A_{121}A_{123} - A_{121}A_{122}A_{123})(1 + A_2A_{11} - A_2 - A_{11})$

5. Determine I_κ^B for each A,

$I_{A_{11}}^B = \frac{\partial P_F(SYS)}{\partial P_F(A_{11})}$
$= (A_2 - 1)(1 - A_{121} - A_{122} - A_{123} + A_{122}A_{123} + A_{121}A_{122} + A_{121}A_{123} - A_{121}A_{122}A_{123})$
$= (0.1 - 1)(1 - 0.1 - 0.1 - 0.1 + 0.01 + 0.01 + 0.01 - 0.001)$
$= -0.6561$

$I_{A_2}^B = \frac{\partial P_F(SYS)}{\partial P_F(A_2)}$
$= 1 + (A_{11} - 1)(1 - A_{121} - A_{122} - A_{123} + A_{122}A_{123} + A_{121}A_{122} + A_{121}A_{123} - A_{121}A_{122}A_{123})$
$= 0.3439$

$I_{A_{121}}^B = \frac{\partial P_F(SYS)}{\partial P_F(A_{121})}$
$= (A_{122} + A_{123} - A_{122}A_{123} - 1)(1 + A_2A_{11} - A_2 - A_{11})$
$= (0.1 + 0.1 - 0.01 - 1)(1 + 0.01 - 0.1 - 0.1)$
$=-0.6561$

Similarly, $I_{A_{122}}^B = I_{A_{123}}^B = -0.6561$
So, I = [0.3439, -0.6561, -0.6561, -0.6561, -0.6561]

6. Sorting array I in decreasing order, we get,
I = [0.3439, -0.6561,-0.6561,-0.6561,-0.6561]
And correspondingly set of attack events is,
S(A)=[A2,A11,A121,A122,A123]

7. Now, we select attack A_2 with highest value of I_κ^B.
From Matrix M, CM ={C3} and COV_ATT= {e}
So OPT = {3} [Since |CM|=1]
S(A) = {a, b, c, d}

In next iteration of while loop, we select attack A at next position of array I. From Matrix M,
CM = {C1} and COV_ATT = $\{A_{11}, A_{121}, A_{122}, A_{123}\}$
So OPT= {C3, C1}
S(A) = {φ}
Hence algorithm stops and the result OPT = {C3, C1}

Now, if we calculate optimal set of countermeasure based on mincut only as specified in [1], we get the same answer. Hence, the algorithm we have proposed is correct but result in our case depends on the way we take up attack events with same probability and also on the way countermeasures with same value of COV_ATT are taken up. This algorithm does not hold for those ACTs where given countermeasures are not sufficient for covering all the attack events and also where the ACTs have repeated events.

VII. CONCLUSIONS AND FUTURE WORK

SYN Flooding attacks are the most common type of DoS attacks and thus it becomes a necessity for every organisation to protect its servers and other resources from it. Hence we have modelled them and found an optimal set of countermeasures using a greedy algorithm proposed in this paper. The algorithm makes use of Birnbaum's Structural Importance Measure to find locally optimal solution, which is a countermeasure covering most critical attack event, at every step. Advantage is that we need not know about probabilities of all the events in ACT which is difficult task in itself. However, this algorithm does not work for the cases where ACTs given countermeasures are not sufficient for covering all the attack events or the ACTs that have repeated events. So in future we are going to work on this algorithm to work for the above mentioned cases also. We are going to implement this algorithm in Network Simulator 2, open source software.

REFERENCES

[1] Arpan Roy, Dong Seong Kim and Kishor S. Trivedi, "Scalable Optimal Countermeasure Selection using Implicit Enumeration on Attack Countermeasure Trees", in Dependable Systems and Networks (DSN), June 2012

[2] Arpan Roy, Dong Seong Kim, and Kishor S. Trivedi, "ACT : Towards unifying the constructs of attack and defense trees" , Security Comm. Networks, pp.1–15, 2011.

[3] B. Schneier, "*Secrets and Lies: Digital Security in a Networked World"*, John Wiley and Sons Inc., 2000.

[4] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, "Optimal security hardening using multi-objective optimization on attack tree models of networks," in *Proc. CCS*. ACM, 2007, pp. 204–213.

[5] S. Bistarelli, M. D. Aglio, and P. Peretti, "Strategic Games on Defense Trees," *LNCS*, vol. 4691, pp. 1–15, 2007.

[6] Birnbaum, Z.W., "On the Importance of Different Components in a Multi-Component system", in MULTIVARIATE ANALYSIS-II , Academic Press, New York, 1969.

[7] Wendai Wang, James Loman, Pantelis Vassiliou,, "Reliability Importance of Components in a Complex System", Reliability and Maintainability, 2004 Annual Symposium – RAMS, Jan 2004

[8] Gan Z, Tang J, Wu P, Varadharajan V., "A Novel Security Risk Evaluation for Information Systems", In FCST, 2007; 67–73.

[9] Saman A. Zonouz, Himanshu Khurana, William H. Sanders, and Timothy M. Yardley, "RRE: A Game-Theoretic Intrusion Response and Recovery Engine", in Critical Infrastructures, April 2009.